Management of the World-Wide Web

Aiko Pras, Harrie Hazewinkel[1], Eric van Hengstum
Centre for Telematics and Information Technology, University of Twente
PO Box 217, 7500 AE Enschede, The Netherlands
pras@cs.utwente.nl - tel.: +31-53-4893778

Abstract

This paper describes the results of a 'proof of concept study', which was sponsored by the Joint Research Centre (JRC) of the European Commission (EC). The subject of the study was to investigate the management capabilities of the World Wide Web (WWW). SNMP was selected as management protocol, but at the time the study started there were no SNMP-WWW MIBs available and research in the area of WWW management was very limited [1]. As a consequence, we had to define our own MIBs and implement our own WWW management applications and agents.
We decided to define three different MIBs. The first one is the server MIB, which holds information on the kind of server, the number of server accesses and the documents provided by the server. The second MIB is the Access Point MIB, which can be used to obtain high level information concerning the transfer of WWW documents over the Internet. The third MIB is the HTTP MIB, and contains detailed information on the operation of the HTTP protocol.
To test the practicability of these MIBs, agents were implemented and installed on a number of WWW servers throughout Europe. The original agent implementation is based on the EMANATE extensible agent package, but recently a public domain version, which is based on Scotty, has also been developed. The results of our study have been submitted to the IETF and are being used within the European DESIRE project.

Keywords: WWW, SNMP, MIB, HTTP, Extensible agent, EMANATE, Scotty

1 Introduction

Within Europe there is an interesting and growing market for earth observation data. Such data is collected from satellites as well as ordinary ground stations that measure things like rainfall and air pollution. To increase the possible use of this data, the European Commission (EC) has formed a programme called the 'Centre for Earth Observation' (CEO). One of the goals of this programme, which is coordinated by the Commission's Joint Research Centre (JRC), is to create a uniform infrastructure via which potential users of earth observation data can interact with the providers of such data. The infrastructure will be build on top of the Internet and will use the 'World-Wide Web' (WWW) as the main application. The infrastructure should be flexible however and allow the addition of alternative applications like the 'File Transfer Protocol' (FTP) and Gopher.

WWW is based on a client-server approach. Upon the mouse-click of a WWW user, the WWW client requests the WWW server to send one or more multi-media documents. After reception of a document, the 'viewer part' of the client interprets the formatting commands that are contained within the document and presents the result to its user. WWW clients are sometimes called 'browsers'; examples come from Netscape, Microsoft and Mosaic.
Well known WWW servers are those from NCSA and Apache. Both servers can be configured such, that they keep copies of all requests and errors in special logfiles.
A special kind of server is the 'gateway' server, which passes requests for documents to special applications like for instance databases. The advantage of gateways is that the contents of docu-

1. Harrie Hazewinkel is currently working for the Joint Research Centre (JRC) of the EC (Ispra, Italy)

ments need not come form static text files, but can be computed from databases at the time the request is made. This guarantees that documents always contain the most recent information.

As shown in Figure 1, WWW can be decomposed into two functional layers. The upper layer, which is located on top of a conceptual document transfer service, is responsible for the formatting and presentation of documents. To accomplish this, it uses the 'HyperText Mark-up Language' (HTML). The lower layer takes care of the transparent[1] transfer of HTML documents over underlying TCP connections. The protocol that is used for this transfer, is the HyperText Transfer Protocol (HTTP) [2].

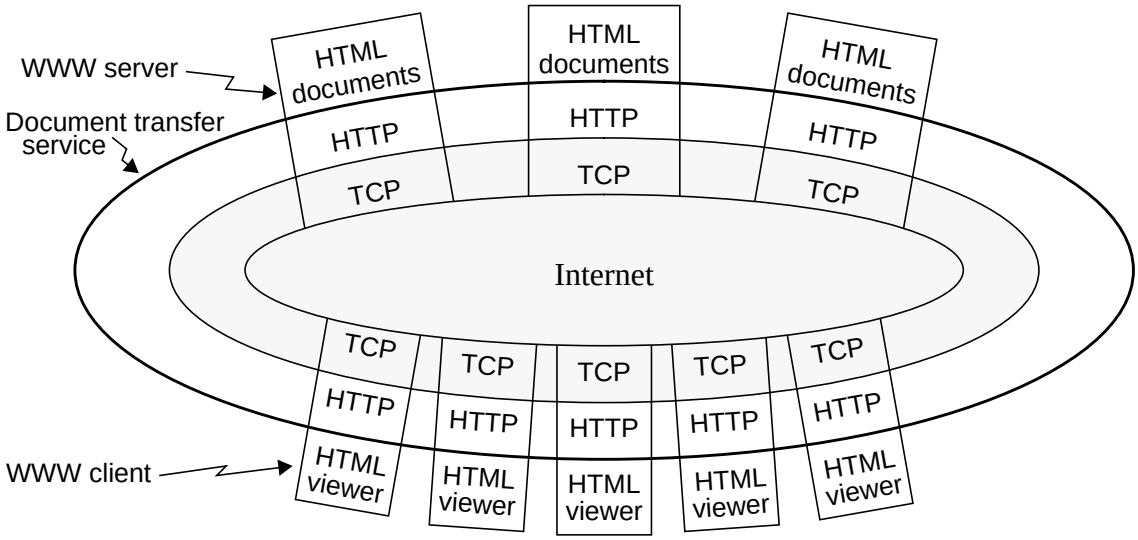

Figure 1: Functional view of the WWW

To investigate the potential management capabilities of this infrastructure, the JRC decided to let ESYS Ltd. (UK) and the University of Twente (The Netherlands) perform a *proof of concept* study [3]. Because the skills of both partners were different, it was agreed that ESYS Ltd. would concentrate on the manager side of the problem and the University of Twente on the agent side. Since this study was the first one of a series, it was not necessary to tackle every possible management problem now; in fact it was sufficient to restrict this first study to the *monitoring* of network performance and server statistics. Because the study would concentrate on Internet technology, the partners also agreed to use SNMP [4] as management protocol.

The emphasis of this paper is on the agent specific aspects of WWW management. Section 2 presents the management architecture and identifies the MIBs that had to be developed; the structure of these MIBs is discussed in Section 3. Section 4 explains the way they were implemented.

2 Management architecture

To enable managers to monitor the WWW servers, a special *Server MIB* (S-MIB) had to be developed. This MIB, which had to be implemented within each server system (Figure 2), contains information like: the server's name, the contact person, the number of times the server has been used, the address of the clients that used this server, the documents that have been retrieved thus far etc. A detailed description of this MIB is provided in Section 3.1.

1. The term 'transparent' means that the contents of documents, and also the HTML code, is not interpreted at this layer.

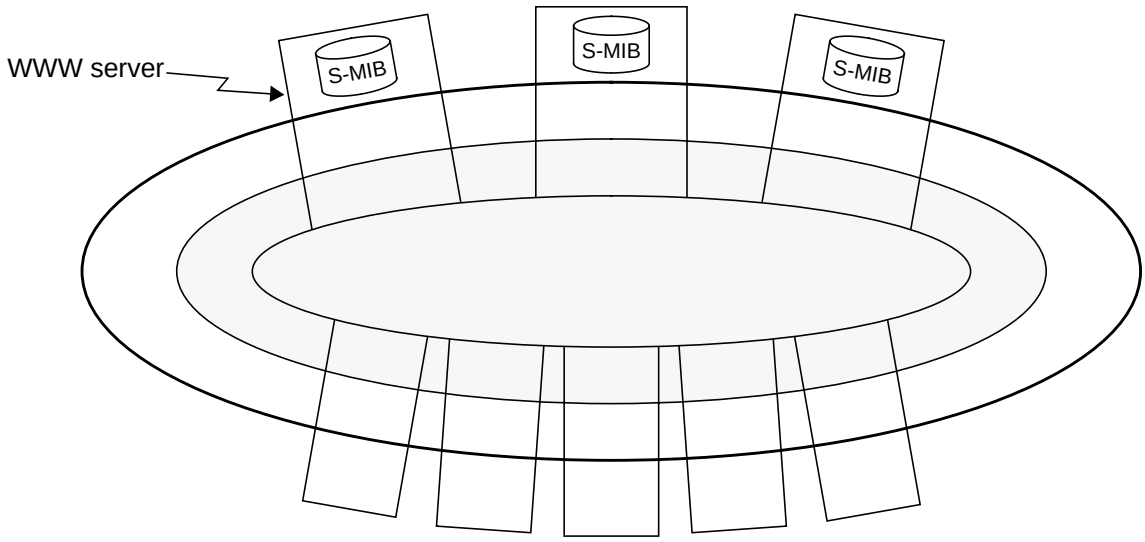

Figure 2: The Server MIB (S-MIB)

Next to monitoring the WWW servers, managers should also be able to monitor the quality under which documents are being transferred. Ideally managers would perform this by interrogating a *central transfer service MIB*, which provides high level information like: how much information is exchanged between each client-server pair, what is the delay, throughput and error rate that each client perceives etc. Unfortunately, this kind of information is not readily available from a central system within the network, but should be computed from rudimentary information which is scattered over all client and server systems. The idea of a service MIB therefore implies the introduction of a kind of *mediation device* (an intermediate level manager), which interacts with the various client and server systems to populate its service MIB. Since a mediation device makes the design more complex and is not essential for this proof of concept study, it was decided to drop the idea of a central service MIB. Instead, it was agreed to develop a special application which would run within the manager system. This application collects the rudimentary information from special *Access Point MIBs* (A-MIB), which were developed within the project and implemented within each client and server system (Figure 3). Section 3.2 provides an overview of this MIB.

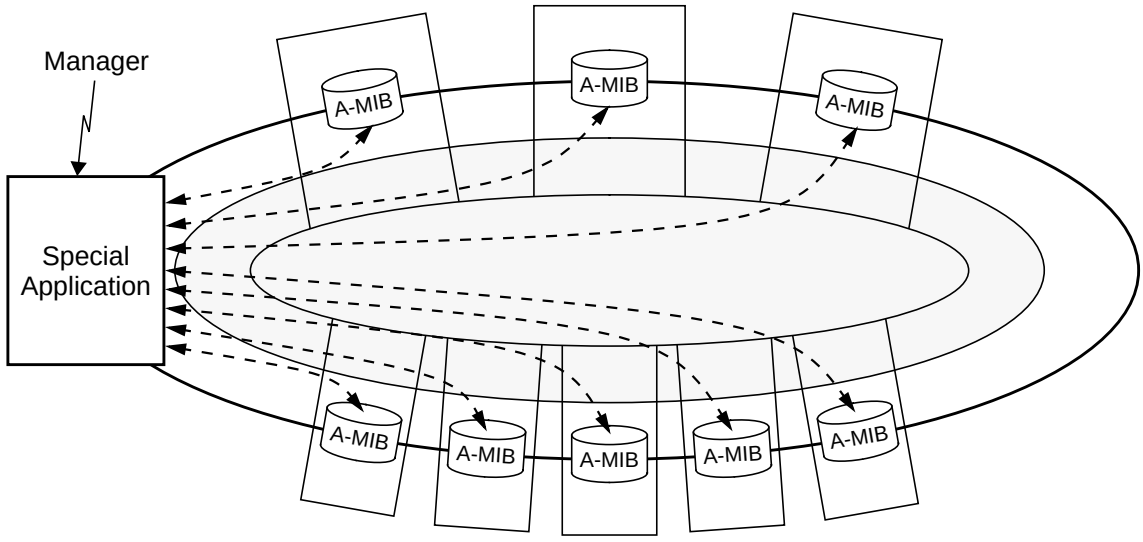

Figure 3: The Access Point MIBs (A-MIB)

Extending each WWW client and server system with an Access Point MIB gives a tremendous amount of work and is not feasible within this proof of concept study. For the time being[1] it was therefore decided to restrict the implementation of the Access Point MIBs to the relatively small number of server systems, which had to be adapted anyway because they had to incorporate the Server MIBs. It is important to understand however, that despite the fact that the Access Point MIB has not been implemented within the client systems, indications of delay values can still be obtained by using the well known ‘ping’ and ‘echo’ functions.

If the manager detects from the Access Point MIBs a problem within the document transfer service, the manager should inspect the various protocol MIBs to find the precise cause of the problem. Management information concerning the IP and TCP protocol can be found in the MIB-II [5], which is usually available within every network system. A MIB for the HTTP protocol was not available at the time the project started, thus we had to develop such MIB (Figure 4) within the project. The structure of this MIB, which is again only implemented within server systems, is discussed in Section 3.3.

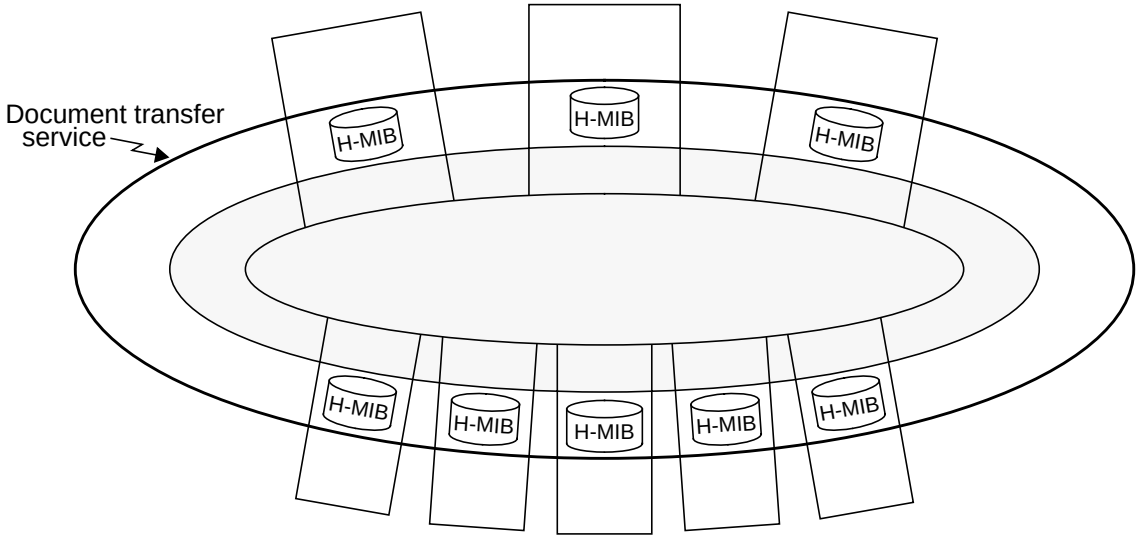

Figure 4: The HTTP MIB (H-MIB)

3 Management information

This Section explains the structure of the three MIBs that were defined within the project:

- the Server MIB (Section 3.1)
- the Access Point MIB (Section 3.2)
- the HTTP MIB (Section 3.3).

1. The ability to monitor the quality of the information transfer may not only be relevant for WWW management, but also for management of other applications. It is therefore believed that the development of Access Point MIBs should take place within a wider context and that the outcome should be standardized within organizations like the IETF. In the long term the result should be that the Access Point MIB is, just as the MIB-II, implemented within every network system (thus also WWW *clients*).

3.1 Server MIB

The Server MIB (Figure 5) provides information about the kind of server (the sGeneral group), the number of server accesses (the sAccess group), the documents which are available at this server (the sDocument group) and the errors that occurred (the sError group).

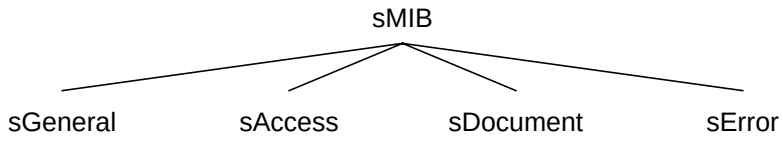

Figure 5: Structure of the Server MIB

The sGeneral group (Figure 6) contains simple variables which holds the server's name, the organisation that operates the server, the contact address of the responsible person, the time the server was last initialized and a variable for the supported media types (e.g. text, pictures, sound and movies). Note that a variable to indicate the location of the server is missing; such variable is already available within the system group of the MIB-II and need not be duplicated. The sGeneral group also includes two tables. The first one, the topicTable indicates the kind of information that is provided by this server. The second one, which is needed in case the server operates as a gateway server, shows which applications (e.g. an Oracle database) are used to generate HTML documents. This table is called the applicationTable and includes the application's name, version, uptime, operational status and errors. In fact the table provides similar information as the application (SYSAPPL) MIB, for which standardization has just been started within the IETF. In the future the applicationTable may therefore be replaced by (parts of) the SYSAPPL-MIB.

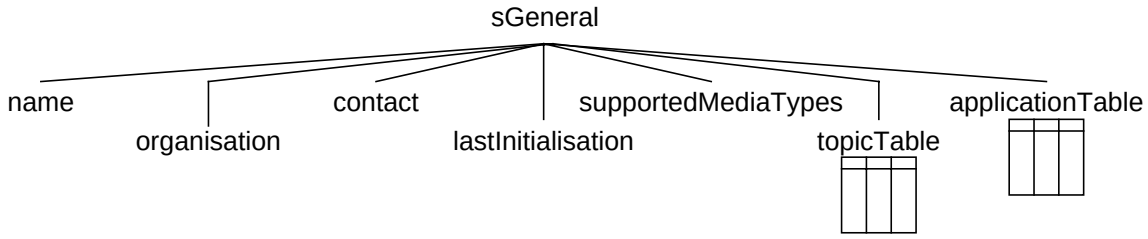

Figure 6: The General group of the Server MIB

The sAccess group informs the manager how the server is being used. Next to a number of tables, the sAccess group provides three summary figures: the total number of accesses, the number of bytes received and the number of bytes transmitted (Figure 7). Although the manager could compute these figures from other MIB information, they were included to avoid the large amount of traffic that must otherwise be exchanged for this computation.
The domainTable contains the number of accesses per user domain and is useful to decide whether and where mirror servers are needed. The lastDaysTable shows how often the server has been accessed during the last couple of days. Like the mostRecentUserTable and the mostFrequentUserTable, the size of this table may be modified by the manager to save RAM space. The two user tables are typical 'TopN' tables; to speed up implementations, the agent will only update the mostFrequentUserTable after it receives a corresponding request from the manager (the manager should set a special *refresh* variable). To avoid superfluous refreshes, an additional variable has been included to show the time the table was last updated.
The sDocument group of the Server MIB maintains information concerning the individual documents that are provided by the server. The group consists of a TopN table, plus associated tableSize and tableRefresh variables that may be set by the manager. The table keeps per document a sepa-

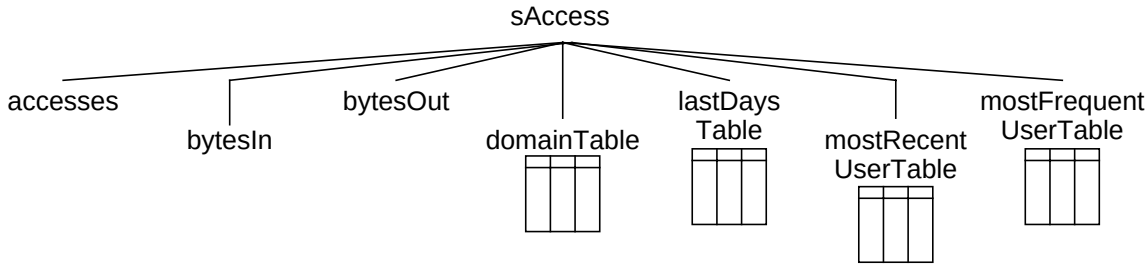

Figure 7: The Access group of the Server MIB

rate row, which holds the document's name, size, type, time of last update, access rights, number of accesses and the number of errors.

The sError group keeps track of the local information retrieval errors that have occurred thus far (information transfer errors are counted elsewhere). To allow definition of additional error types, the group is organised as a table. For each error type the table maintains an independent row, which holds a description of the error, the number of such errors and the last occurrence of that error.

3.2 Access Point MIB

The Access Point MIB consists of a serviceStatistics group and a qualityOfService group. The serviceStatistics group at the client side keeps track of the number of service requests and confirms; at the server side it counts the number of service indications and responses (Figure 8).

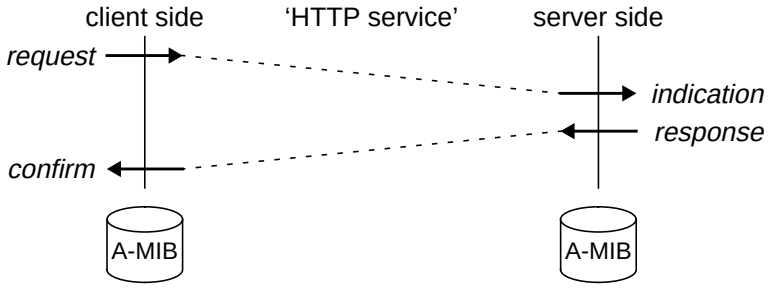

Figure 8: Idea of the Access Point MIB (A-MIB)

The qualityOfService group consists of a delay table, a throughput table and a time-out variable (Figure 9). The two tables are indexed by the address of the remote system. At the client side the delay can be determined by measuring the interval between the occurrence of the request and the occurrence of the confirm; at the server side delay values can be estimated by using ICMP's *ping* or UDP's *echo* function [6][7]. The time-out variable counts how many confirms got lost.

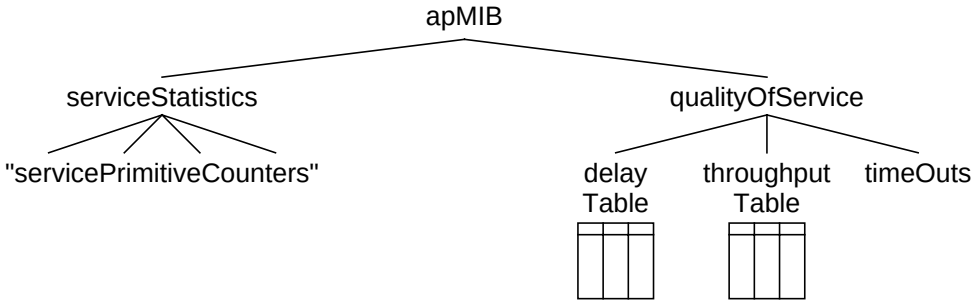

Figure 9: Structure of the Access Point MIB

3.3 HTTP MIB

The HTTP MIB (Figure 10) consists of three groups: one for general information (the httpSystem group), one for statistical information (the httpStatistics group) and one for errors (the httpTime-Outs group).
The system group contains information such as the vendor of the HTTP entity, the HTTP version number, the address of the entity and its uptime. Since the same MIB definition should be applicable to clients as well as servers, the system group contains a variable to indicate the entity's role. To allow the coexistence of multiple HTTP entities within a single system, all information within the HTTP MIB is structured into tables; rows within the tables that share the same index refer to the same HTTP entity.

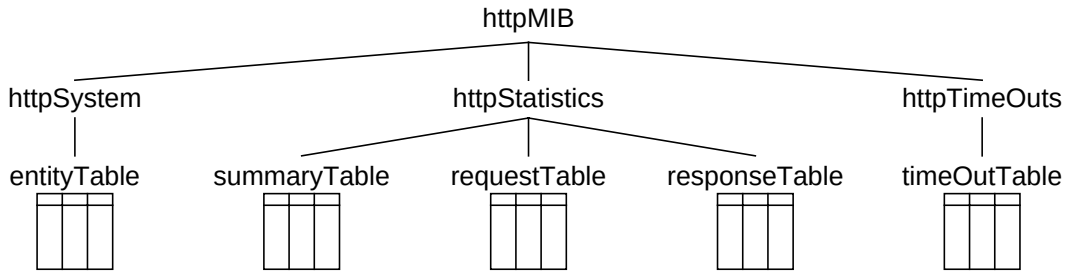

Figure 10: Structure of the HTTP MIB

The summary table within the statistics group provides a quick overview of the entity's performance; it shows the total number of requests and responses that have been generated and received (depending on the entity's role), the number of incoming and outgoing bytes, as well as the number of errors. Although the manager could compute part of this information from the other tables, the summary table was included to reduce SNMP network load.

Detailed information, such as timestamps indicating the last occurrence of each PDU type, can be found in the request and response tables. These tables have for each possible request / response type a separate entry and can be easily extended to accommodate new types.
The timeOutTable is a TopN table and contains per remote system the number of timeouts as well as the time of the last time-out. The size of the table can be specified by the manager.

4 Implementation

After the definition of the MIBs was complete, the MIBs were implemented as part of an SNMP agent. To test the MIBs, the agent was incorporated into several WWW servers, all of them being UNIX systems.

A common way of implementing a SNMP agent within a UNIX environments, is to use a separate UNIX process which interacts with the process that is being managed: the WWW server process. The interaction between server and agent processes is usually realized by a two-way Inter-Process Communication (IPC) mechanism (left side of Figure 11). Such mechanism allows the agent to *monitor* the server's behaviour by fetching information from the server, and to *influence* the server's behaviour by changing its variables.
In principle, the IPC strategy allows the SNMP agent to exercise complete control over the server. Unfortunately, the server implementations that existed at the time the project started did not include such IPC capabilities. This implies that we had to modify existing server software, which was clearly undesirable.

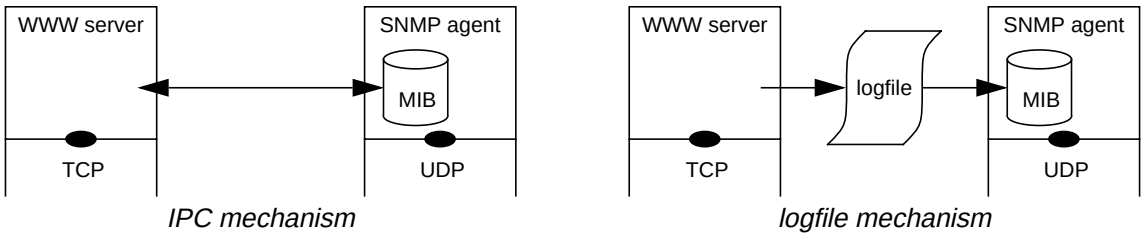

Figure 11: Different implementation approaches

However, within our proof of concept study the requirement was to *monitor* the server's behaviour; *influencing* the behaviour was not a requirement. It is therefore sufficient to realise a *one-way* information exchange mechanism from server to agent, a full-fledged IPC mechanism is not yet required. An elegant way to implement this one-way exchange, is to use the logfiles which are maintained by each WWW server (right side of Figure 11). The format of these logfiles is described in a de facto standard [8], which means that the same agent implementation can be used to manage servers from different vendors.
After the advantages and disadvantages of both approaches were understood (Figure 12), we decided to adopt the logfile approach.

method	advantage	disadvantage
IPC approach	• complete control over server (set operations are possible)	• server software need to be modified
logfile approach	• no need to modify server software • can be used with different servers	• server can only be monitored (set operations are not possible)

Figure 12: Comparison between IPC and logfiles

Next to the three WWW MIBs, the SNMP agent should also support the MIB-II to allow management of the lower layer protocols, such as TCP and IP. It may be expected that in the future other MIBs must be supported too. It was therefore decided to implement the agent as an *extensible agent* [9]. With such agents, the MIB specific functions are implemented within so-called *subagents*, and the common functions, such as the Basic Encoding Rules (BER), are implemented within a central *master agent*. The master agent and the various subagents are separate UNIX processes. In case a new MIB must be added, it is only necessary to develop or buy a new subagent; the other subagents, as well as the master agent, remain intact.

To speed up our work, it was decided to use the EMANATE extensible agent package from SNMP Research [10]. This package includes a master agent, several subagents (for instance for MIB-II) and a subagent development kit, which was used to implement the WWW-subagent (Figure 13). This subagent monitors the tail of the logfiles, where the server writes its new information. Since the Server MIB (S-MIB), Access Point MIB (A-MIB) and HTTP MIB (H-MIB) are populated from the same logfiles, all WWW MIBs are implemented within a single subagent.
The implementation of the WWW-subagent has about 12000 lines of C-code, of which 50% is generated by the EMANATE subagent development kit. The size of the subagent is 1.2 Mbyte, which is slightly more than the master agent (1.1 Mbyte). The agent has been tested with logfiles of 5 Mbyte and could return around 8 variables per second.

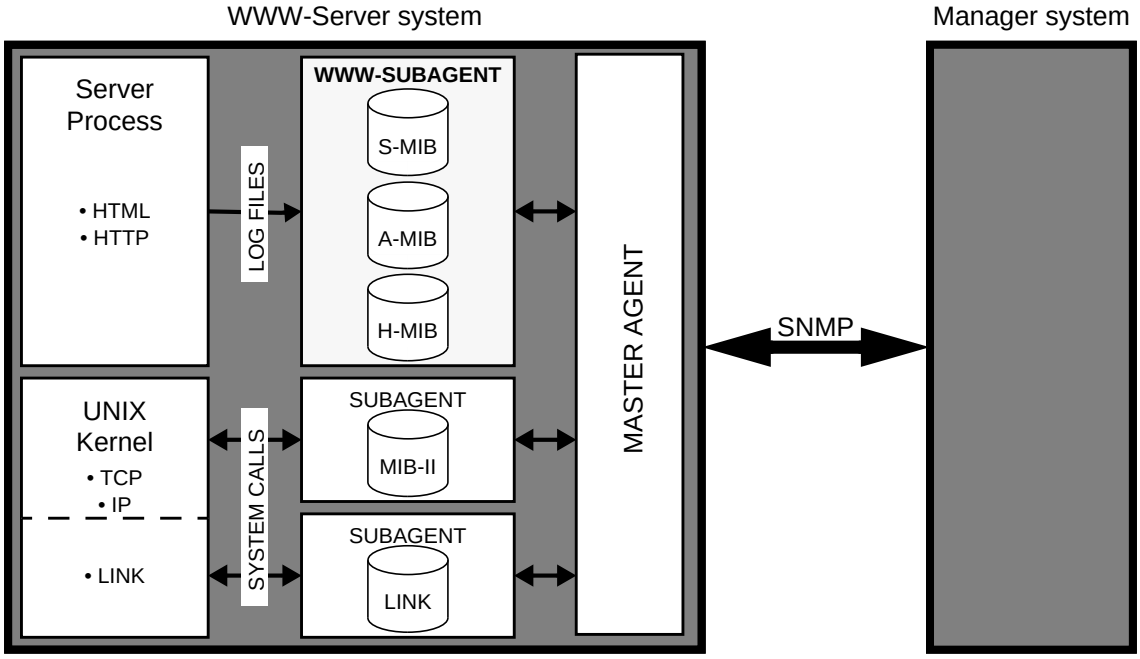

Figure 13: Structure of the implementation

5 Conclusions

The agent has been installed on a number of European servers for Earth Observation Data and is currently being tested. To allow others outside the CEO programme to take advantage of our work, we have also developed a public domain WWW manager and agent. The manager is based on TKINED [11] and can, together with the Scotty based agent [12], be downloaded from our Simpleweb server [13].

The Server MIB, Access Point MIB and HTTP MIB were presented as Internet drafts to the IETF, where a particular interest existed in the latter MIB. The description of our MIB was therefore used as one of the starting points for the HTTP MIB working group [14]. To get an agreement on an HTTP MIB, the MIB as presented in this paper will require a number of changes, for instance to accommodate the results of the other IETF group which has recently started on the related topic of an Application MIB (SYSAPPL-MIB).

The MIBs and prototypes that are presented in this paper, will also be used within the European DESIRE project; which is a TELEMATICS project within the fourth framework programme of the EC.

Acknowledgements

We would like to thank the CEO programme of the JRC, in particular Rui Meneses, who made this proof of concept study possible. We would also like to thank Mark Gamble of ESYS Ltd., and the members of the HTTP mailing list, in particular Carl Kalbfleisch and Jürgen Schönwälder.

References

[1] Picoto C., Veiga P.: "*Management of a WWW server using SNMP*", University of Lisbon, Portugal, 1995

[2] Berners-Lee, T., Fielding R. and Frystyk, H.:, "*Hypertext Transfer Protocol -- HTTP/1.0*", RFC 1945, 1996.

[3] Hazewinkel H., Hengstum F.P.H. v., Pras A.: "*Results of the CEO Project - WWW Management*", CTIT Technical Report Series No. 96-18, ISSN 1381-3625, University of Twente, The Netherlands, 1996

[4] Case J.D., Fedor M., Schoffstall M.L., Davin C.: "*Simple Network Management Protocol (SNMP)*", RFC 1157, 1990.

[5] McCloghrie K., Rose M.: "*Management Information Base for Network Management of TCP/IP-based internets (MIB-II)*", RFC 1213

[6] Postel J.: "*User Datagram Protocol (UDP)*", RFC 768, 1980.

[7] Postel J.: "*Internet Control Message Protocol (ICMP)*", RFC 792, 1981.

[8] Common Logfile Format: http://www.w3.org/hypertext/WWW/Daemon/User/Config/Logging.html

[9] Linde H. v.d.: "*Evaluation and new Specification of the Extensible Agent Protocol*", M.Sc. thesis, University of Twente, 1995

[10] SNMP Research: "*EMANATE Subagent Development Kit*", SNMP Research International Inc., Knoxville, Tennessee, 1994

[11] Schönwälder J., Langendörfer H. "*INED - An Application Independent Network Editor*", In Proc. World Conference on Tools and Techniques for System Administration, Networking and Security, Arlington, 1993

[12] Schönwälder J., Langendörfer H.: "*Tcl Extensions for Network Management Applications*", In Proc. 3rd Tcl/Tk Workshop, page 279-288, Toronto, 1995

[13] The Simpleweb: http://wwwsnmp.cs.utwente.nl/

[14] HTTP Mailing list: http://www.onramp.net/~cwk/http-mib/email.html